Software Requirements Specification

For

GTA Application Portal

November 27, 2023

Version 2

Prepared by:

Brandon Piotrowski

**Table of Contents**

[**1 Introduction 4**](#)

[1.1 Overview 4](#_3znysh7)

[1.2 Goals and Objectives 4](#)

[1.3 Scope 4](#)

[1.4 Definitions 4](#)

[1.5 Assumptions 5](#)

[**2 General Design Constraints 5**](#)

[2.1 Product Environment 5](#_17dp8vu)

[2.2 User Characteristics 5](#)

[2.3 Mandated Constraints 6](#)

[2.4 Potential System Evolution 6](#)

[**3 Nonfunctional Requirements 6**](#)

[3.1 Operational Requirements 6](#)

[3.2 Performance Requirements 6](#)

[3.3 Security Requirements 6](#)

[3.4 Other Quality Attributes 6](#)

[3.5 Documentation and Training 6](#)

[3.6 External Interface 6](#)

[3.6.1 User Interface 6](#)

[3.6.2 Software Interface 7](#)

[**4 System Features 7**](#)

[4.1 Student User Features 7](#)

[4.1.1 Use Case: 1 7](#)

[4.1.2 Use Case: 2 7](#)

[4.1.3 Use Case: 3 7](#)

[4.1.4 Use Case: 4 8](#_mgu04pv2n788)

[4.1.5 Use Case: 5 8](#_hwnp9emeuu7e)

[4.2 Administrative User Features 9](#)

[4.2.1 Use Case: 6 9](#_3o7alnk)

[4.2.2 Use Case: 7 9](#)

[4.2.3 Use Case: 8 9](#_ihv636)

[4.2.4 Use Case: 9 10](#_ro3hci86qeke)

[4.2.5 Use Case: 10 10](#_5fjv6l6n4qgy)

Revision History

| **Version** | **Date** | **Name** | **Description** |
| --- | --- | --- | --- |
| 1 | 11/27/2023 | Brandon Piotrowski | Initial Document |
| 2 | 12/8/2023 | Brandon Piotrowski | Final Revision |

# 1 **Introduction**

## ***Overview***

The GTA Application Portal is a website available to UMKC students who want to apply to be a grader or lab instructor for specific classes at the university. For students, the website will provide access to a list of jobs they can apply to, a list of their applications they have submitted, and their profile page for editing their information. For administrators, the website will provide access to everything that a student user can access and an administration page. The administration page allows an administrator to create new job listings, edit existing job listings, modify student applications, open and close student applications, and view student users.

The GTA Application Portal’s requirements will be explained in this document. The goals, objectives, scope, definitions and assumptions are described in the introduction. General design constraints such as the product environment, user characteristics, mandated constraints, and the potential evolution of the system are given in the following section. Non-functional requirements are also given with functional requirements following those.

## ***Goals and Objectives***

The main objective of this project is to allow students a way to apply for grading and lab assistant positions for specific classes held at the university. The GTA Application Portal is expected to:

1. Provide an intuitive interface for students to apply to different positions, manage their applications, and change their information
2. Provide administrators with an interface to efficiently manage student applications, edit job listings, post jobs, and manage user accounts
3. Store user information in a database
4. Have an easy-to-understand layout and simple functionality
5. Follow similar UI design to the UMKC website

## ***Scope***

The GTA Application Portal will provide students a website with the ability to apply to different grader and lab instructor positions for UMKC courses. Students will also be able to manage their current applications and modify their account information. When creating an account students will also be able to add courses they have taken to help with the application process. Administrators will be able to add and modify job listings, manage user applications, and manage users.

## ***Definitions***

**GTA Application Portal -** The website that is being described in this document

**User -** A person who interacts with the GTA Application Portal website

**Unauthorized User -** A user who is not currently logged in with an account on the GTA Application Portal website

**Administrator** - A user who has authoritative access to the data stored in the database of the GTA Application Portal website. This user also has deeper access to the underlying systems of the website

**Student** - A user who is currently studying or pursuing an education at UMKC, and is logged in with minimal access to the underlying systems of the GTA Application Portal website

**Use Case** - describes a goal-oriented interaction between the system and an actor. A use case may define several variants called scenarios that result in different paths through the use case and usually different outcomes.

## ***Assumptions***

It is assumed that the user running the program has access to a compatible terminal or IDE that can run Flask programs. It is also assumed that the user has access to the internet and a web browser.

# **General Design Constraints**

## ***Product Environment***

The GTA Application Portal will be a website that is accessible from any browser. The website will be hosted through a web server. The website will also utilize a SQLite database to access user information.



## ***User Characteristics***

**Student User:** A UMKC student, who has access to an internet browser. Students are currently working through their college education and are likely knowledgeable about applying for different positions and using a website with an account.

**Administrator User:** A UMKC faculty member in charge of interviewing and hiring students for different positions at the university. They are likely to know how to use different UMKC systems and technologies.

**Unauthorized User:** A person who does not have an account or is not logged in to the website. They are likely to be knowledgeable of how to create an account for a website.

## ***Mandated Constraints***

The website will be created with Python using the Flask web application framework. SQLite will be the database used for storing user information. The website will be hosted locally but will be designed in a way that it can be hosted on a server.

## ***Potential System Evolution***

Currently the website is hosted locally but the system will be designed in a way that it can eventually be hosted on a server as an independent website.

# **Nonfunctional Requirements**

## ***Operational Requirements***

The website shall be accessible from any computer with an internet connection and a web browser.

## ***Performance Requirements***

A drastic increase to the number of user accounts and applications stored in the database should not increase the retrieval time of data by more than 1 second.

## ***Security Requirements***

Student users shall only have access to applications they have submitted when an administrator opens the application for modification. Also, student users shall only have access to their user information, no other users information. Any administrative user shall have access to all submitted applications from any student user, all job listings posted by any other administrator, and all student account information.

## ***Other Quality Attributes***

An unauthorized user shall have access to view all job listings, but will always be redirected to the log in or register page when clicking an apply button for a job.

## ***Documentation and Training***

The GTA Application Portal will be a standalone website with no provided documentation, and no training needed to use the website.

## ***External Interface***

### **User Interface**

The user interface will follow similar patterns to other UMKC websites. It will feel simple and minimalistic while also maintaining the characteristics of a UMKC website. The user interface will also follow similar design concepts and have a familiar layout to other commonly used websites. 95% of users will be able to use the website with minimal or no issues, and without training.

### **Software Interface**

The local or remote server being used will serve as the interface between the GTA Application Portal website and the SQLite database.

# **System Features**

## Student User Features

### Use Case: 1

**Description: Register Student User Account**

Cost: High

Value: High

Basic Path

1. Unauthorized user clicks “Register” on the top right corner of the web page
2. System prompts user with multiple input boxes for the user’s information
3. Unauthorized user fills in the inputs with their information
4. Unauthorized user clicks the submit button
5. System returns to the login page

### **Use Case:** 2

**Description: Student User Login**

Cost: High

Value: High

Basic Path

1. Unauthorized user clicks “Log In” on the top right corner of the web page
2. System prompts user for their email and password
3. Unauthorized user fills in their email address and password
4. Unauthorized user clicks the submit button
5. System returns to the home page with job listings

Alternate Path

1. Unauthorized user clicks “Apply” button for a job listing on the home page
2. System redirects to the login page
3. System prompts user for their email and password
4. Unauthorized user fills in their email address and password
5. Unauthorized user clicks the submit button
6. System returns to the home page with job listings

### Use Case: 3

**Description: Apply for Job Listing**

Cost: High

Value: High

Basic Path

1. Following login (Use Case 2 Step 6): Student clicks on “Apply” button for a job listing on the home page
2. System redirects to an application page
3. System prompts student with multiple input boxes to submit their information, but autofills all that can be gathered from the student’s account information
4. Student fills in other required information
5. Student clicks the submit button
6. System notifies the student that the application has been submitted
7. System returns to the home page with job listings

### Use Case: 4

**Description: Edit a Submitted Job Application**

Cost: High

Value: High

Basic Path

1. Following login (Use Case 2 Step 6): Student clicks on the “My Applications” button towards the top left corner of the web page
2. System redirects to the my applications page, displaying all of the student’s submitted applications
3. If the application is open for editing, the student clicks the edit button. (Otherwise, the student will be notified it is not currently editable)
4. System redirects to the application page for editing
5. (Steps 3-6 of Use Case 3)
6. System returns to the my applications page.

### Use Case: 5

**Description: Edit User Account Information**

Cost: Medium

Value: High

Basic Path

1. Following login (Use Case 2 Step 6): Student clicks on their name in the top right corner of the web page
2. System redirects to the user account page, displaying a box of all of the user information and multiple input boxes below that
3. The student enters information into the input boxes
4. Student clicks the submit button
5. System changes the previous information with the new information provided by the student
6. System notifies the student of these changes

## Administrative User Features

### Use Case: 6

**Description: Administrative User Login**

Cost: High

Value: High

Basic Path

1. Unauthorized user clicks “Log In” on the top right corner of the web page
2. System prompts user for their email and password
3. Unauthorized user fills in their email address and password
4. Unauthorized user clicks the submit button
5. System returns to the home page with job listings

Alternate Path

1. Unauthorized user clicks “Apply” button for a job listing on the home page
2. System redirects to the login page
3. System prompts user for their email and password
4. Unauthorized user fills in their email address and password
5. Unauthorized user clicks the submit button
6. System returns to the home page with job listings

### **Use Case:** 7

**Description: Create New Job Listing**

Cost: High

Value: High

Basic Path

1. Following login (Use Case 6 Step 6): Administrator clicks on the “Administration” button towards the top left of the web page
2. System redirects to the administration page, displaying a button to create a new job listing, a list of job listings with an edit and open/close button, a list of submitted student applications with an edit and open/close button, and list of student accounts
3. Administrator clicks on the “Create New Job Listing” button
4. System prompts administrator with a drop down for the job role, a drop down for the course, and a toggle for certification requirements
5. Administrator fills in prompts
6. Administrator clicks submit
7. System notifies administrator that the job has been created
8. System redirects back to the administration page

### Use Case: 8

**Description: Edit Existing Job Listing**

Cost: High

Value: High

Basic Path

1. Following access of the administration page (Use Case 7 Steps 1-2): Administrator clicks on the edit button for a job listing
2. System redirects to the corresponding job creation page related to the existing job listing
3. System prompts the administrator with the job role, course name, and certification required inputs
4. Administrator makes the needed changes to the information
5. Administrator clicks submit button
6. System notifies administrator of successful edit
7. System redirects to the administration page

### Use Case: 9

**Description: Open/Close a Submitted Student Job Application**

Cost: Medium

Value: High

Basic Path

1. Following access of the administration page (Use Case 7 Steps 1-2): Administrator clicks on the open/close button for a job listing
2. System changes the status listing of the application

### Use Case: 10

**Description: Edit User Account Information**

Cost: Medium

Value: High

Basic Path

1. Following login (Use Case 6 Step 6): Administrator clicks on their name in the top right corner of the web page
2. System redirects to the user account page, displaying a box of all of the user information and multiple input boxes below that
3. The administrator enters information into the input boxes
4. Administrator clicks the submit button
5. System changes the previous information with the new information provided by the administrator
6. System notifies the administrator of these changes